



US Utilities Stock Forecast

from

H@p.py Group

Group Members

6610110214 Peeranat Pathomkul

6610110425 Keetasin Kongsee

6610110475 Natdanai Chookool

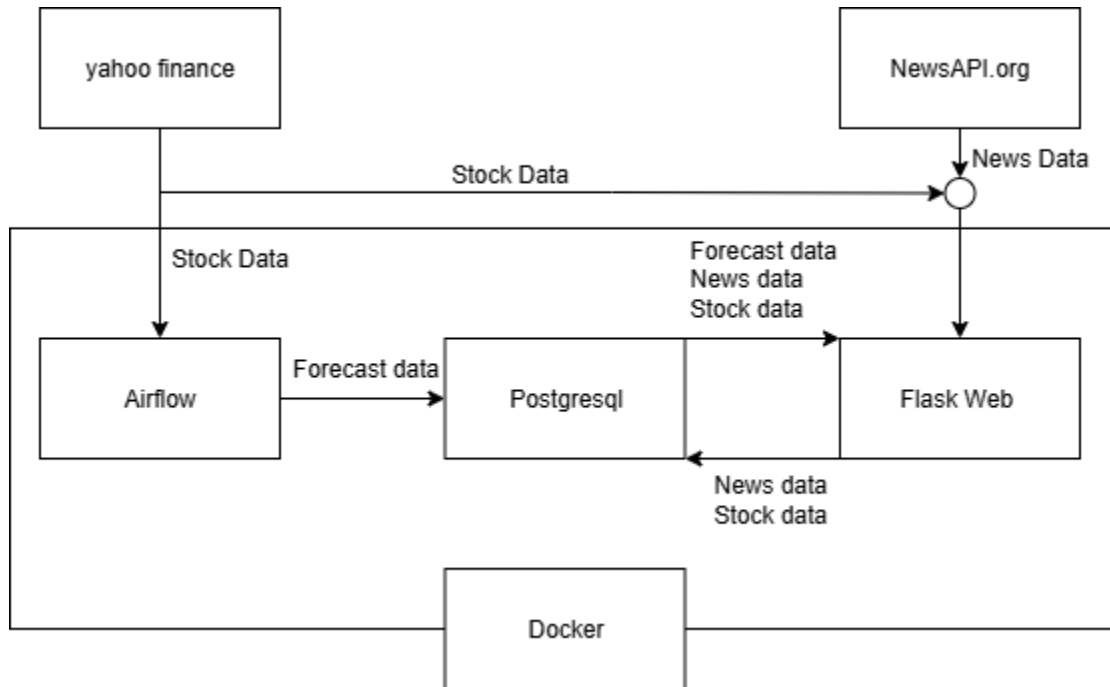
241-353 AI Ecosystems (1/2025)

Prince of Songkla University

GitHub Repository: <https://github.com/Keetasin/US-Utilities-Forecast>

Part A: Time Series Analysis and Forecasting Model (10 Marks)

1. Overview picture of your project



This figure illustrates the overall system architecture of the project.

- Retrieve stock data from the Yahoo Finance API to perform forecasts within an automated Airflow pipeline that runs every business day at 19:30 and store the forecast results in a PostgreSQL database.
- Automatically retrieve stock data from the Yahoo Finance API every minute during market hours, as well as news data from NewsAPI.org every business day at 19:00 and store all data in a PostgreSQL database.
- Display all results through a Flask web application.
- All components operate within a Docker environment.

Data Sources

- Yahoo Finance API provides real-time and historical stock price data.
- NewsAPI.org supplies financial news articles related to each company for contextual insights.

Airflow

- Automates data retrieval from Yahoo Finance.
- Executes forecasting models (ARIMA, SARIMA, SARIMAX, LSTM).
- Saves the forecast results into PostgreSQL for later use.

PostgreSQL










- Stores three main datasets: stock data, forecast results, and stock news.
- Acts as a bridge between the data pipeline (Airflow) and the web interface (Flask).

Flask Web Application

- Retrieves data from PostgreSQL and visualizes the data.
- Provides users with an interactive interface for stock trend analysis and model comparison.

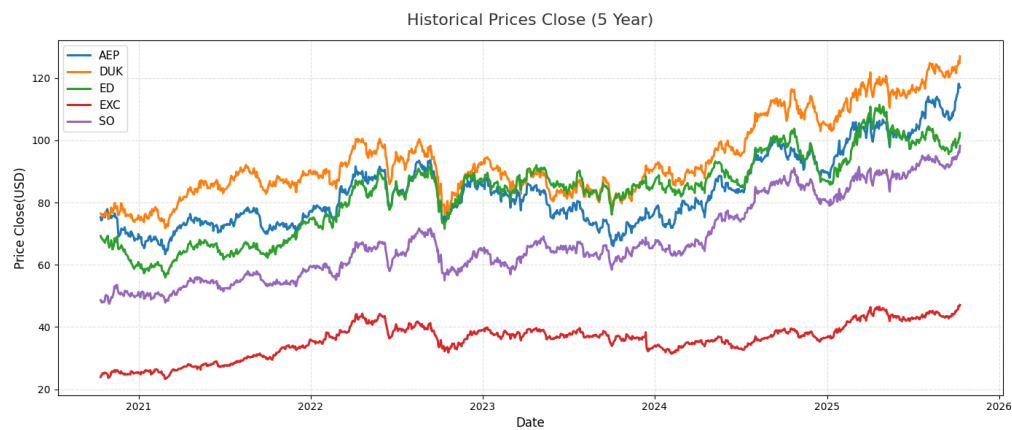
Docker Environment

- All components (Airflow, Flask, PostgreSQL) are containerized using Docker.

| Name ↑ | Image | Status | CPU (%) | Port(s) |
|---|---------------------------------------|---------------|---------|---|
| <div>  mini-project </div> | | Running (4/5) | 4.34% | |
| <div>  airflow-init-1 e4470a0e2695 <div></div> </div> | custom-airflow:latest | Exited | 0% | |
| <div>  postgres-1 143512a67c2f <div></div> </div> | postgres:13 | Running | 1.38% | 5432:5432  |
| <div>  scheduler-1 7ebc09094b8b <div></div> </div> | custom-airflow:latest | Running | 2.77% | |
| <div>  web-1 3819037f08df <div></div> </div> | mini-project-web | Running | 0.03% | 5000:5000  |
| <div>  webserver-1 e58b7284f0f1 <div></div> </div> | custom-airflow:latest | Running | 0.16% | 8080:8080  |

| Service | Description |
|--------------|---|
| postgres | PostgreSQL database for Airflow metadata and application data |
| airflow-init | Initializes Airflow DB and creates an admin user |
| webserver | Airflow web UI (http://localhost:8080) |
| scheduler | Airflow scheduler that executes DAGs |
| web | Flask web application (http://localhost:5000) |

2. Show Exploratory Data Analysis (EDA)



Visualization of historical closing prices for five U.S. utility stocks AEP, DUK, SO, ED, and EXC over the past five years.

– Show Descriptive data

| Ticker | AEP | DUK | ED | EXC | SO |
|--------|-------------|-------------|-------------|-------------|-------------|
| count | 1255.000000 | 1255.000000 | 1255.000000 | 1255.000000 | 1255.000000 |
| mean | 83.610231 | 93.877858 | 83.163959 | 35.796146 | 67.444957 |
| std | 11.732576 | 13.051761 | 12.873581 | 5.470784 | 13.048461 |
| min | 63.398174 | 71.707932 | 55.941063 | 23.265762 | 47.457859 |
| 25% | 74.256844 | 84.841614 | 73.360561 | 33.087013 | 57.360233 |
| 50% | 80.536423 | 89.870697 | 85.176811 | 36.617470 | 64.643059 |
| 75% | 90.435509 | 100.503025 | 90.564362 | 39.018652 | 76.860516 |
| max | 118.190002 | 125.559998 | 111.421577 | 46.790001 | 96.419998 |

The table displays the descriptive statistics for the daily closing prices of five utility sector stocks

- **Count:** All stocks have the same number of data points, 1255.
- **Mean:** DUK has the highest average price (93.88), while EXC has the lowest (35.8). The means for AEP, ED, and SO are clustered around the 67-84 range.
- **Standard Deviation:** This measures the volatility or spread of the prices. DUK (13.05) and SO (13.05) show the highest volatility, whereas EXC (5.47) exhibits the lowest.
- **Min:** EXC has the lowest recorded price (23.27), and DUK has the highest minimum price (71.71).
- **Max:** DUK also reached the highest price overall (125.56), and EXC the lowest maximum price (46.79).
- **Percentiles (25%, 50%, 75%):** These indicate the distribution of the data. For instance, the median (50%) for DUK (89.87) is much higher than that of EXC (36.62).
The Interquartile Range (75% - 25%) confirms that EXC has the tightest price range, while DUK and AEP have wider ranges.

In summary:

- DUK has the highest average price and the highest maximum price, but also one of the highest volatilities.
- EXC has the lowest average price, the lowest minimum price, and by far the lowest volatility (most stable price).
- The other three stocks (AEP, ED, and SO) fall in the mid-range for price and volatility.

- Demonstrate how to check and manage missing values and Outliers.

```
=== Missing Value Count (Before) ===  
Ticker  
AEP      0  
DUK      0  
ED       0  
EXC      0  
SO       0  
dtype: int64
```

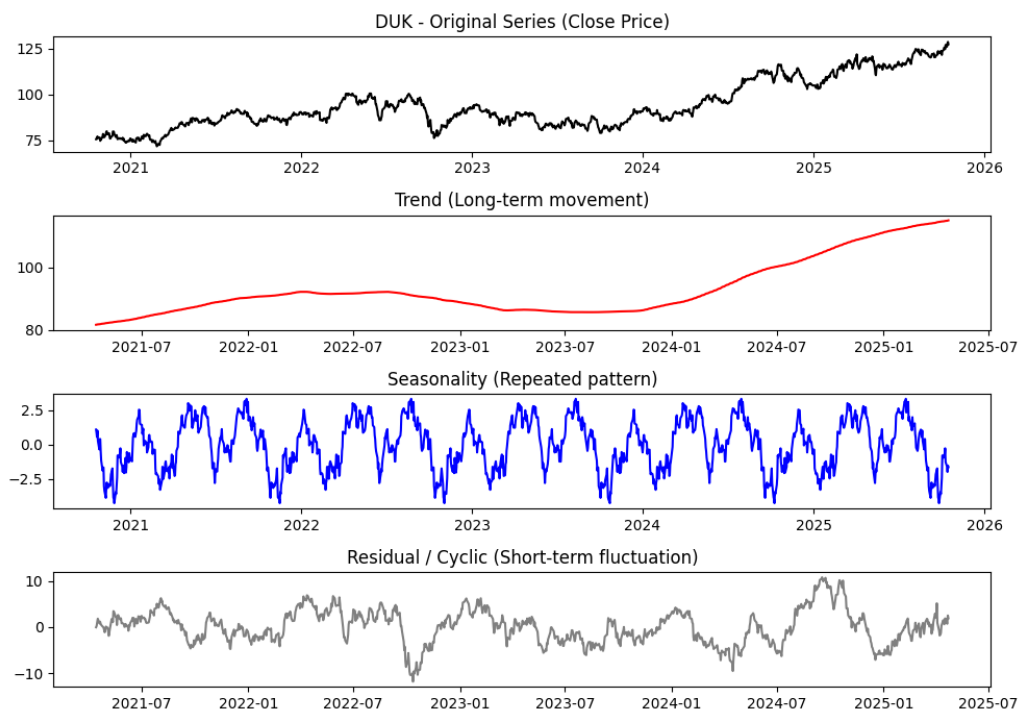
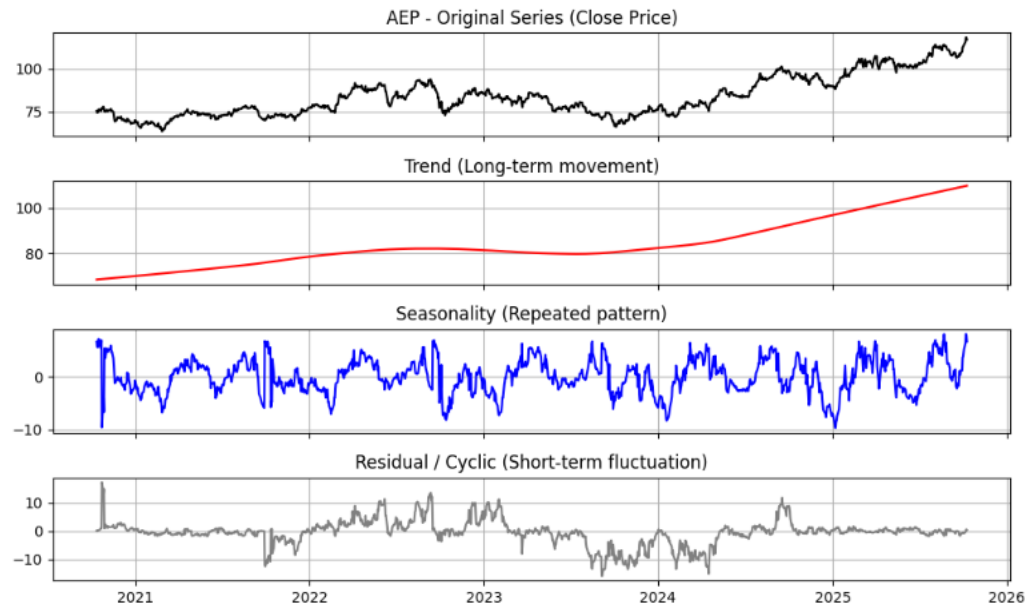
No missing values were found because Yahoo Finance provides cleaned data.

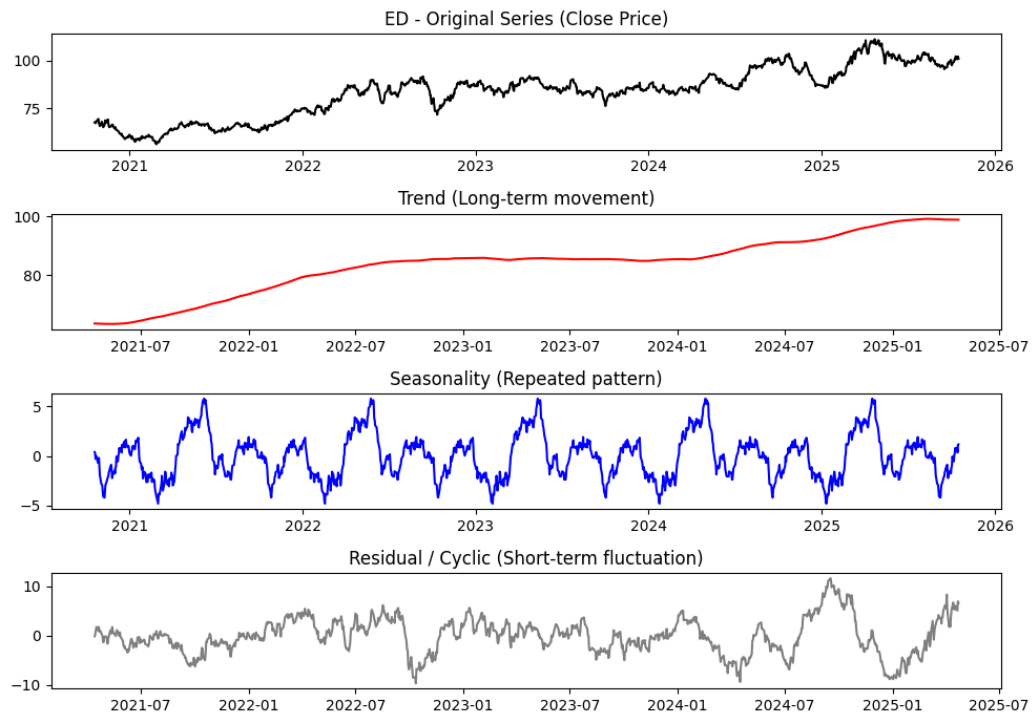
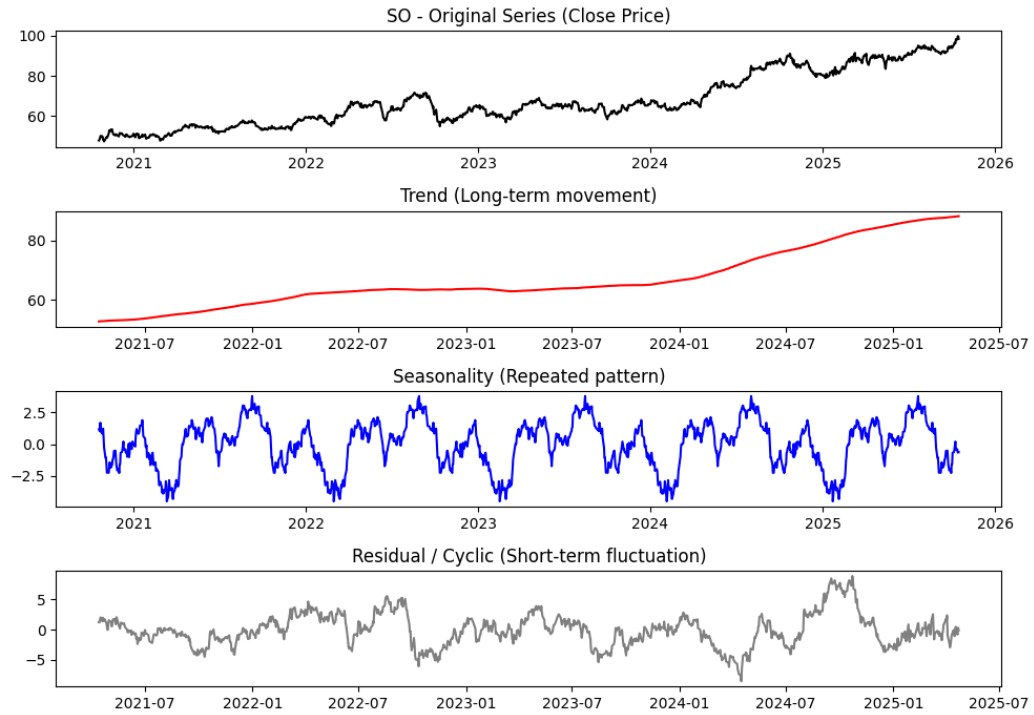
```
=== Outlier Count ( $|Z| > 3$ ) ===  
AEP      13  
DUK       9  
ED       11  
EXC      17  
SO       13  
dtype: int64
```

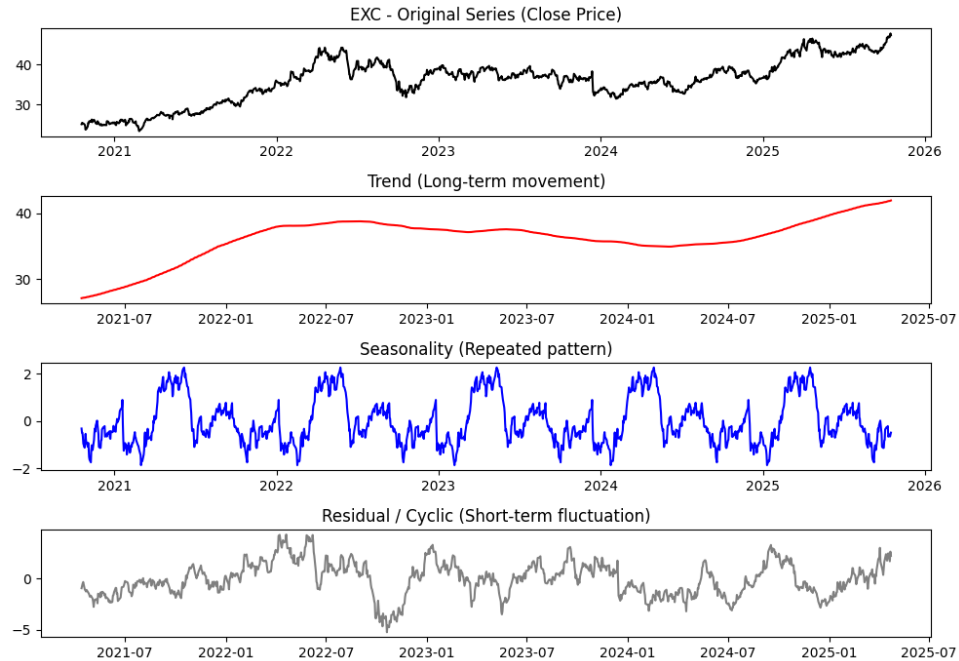
Outliers were analyzed but not removed since they represent real market fluctuations.

3. Data Preprocessing

- Show how to check Trend, Seasonality, and Cyclic







The time series of closing prices for five U.S. utility stocks — AEP, DUK, SO, ED, and EXC — were decomposed using the `seasonal_decompose()` function to identify their underlying patterns.

The decomposition results reveal the following characteristics:

- Trend: All stocks exhibit a clear upward long-term trend, indicating a steady price growth between 2020 and 2025.
- Seasonality: A repeated yearly pattern is observed in all series, showing cyclical movements likely associated with seasonal energy demand or financial cycles.
- Cyclic / Residual: Each stock shows short-term fluctuations due to market volatility, news, or external economic factors.

| Symbol | Trend Behavior | Seasonality Pattern | Cyclic / Residual Observation |
|--------|--|----------------------------|--|
| AEP | Upward trend with gradual increase | Moderate annual pattern | Short-term fluctuations during 2022–2024 |
| DUK | Steady upward trend | Clear yearly repetition | Small fluctuations; stable residuals |
| SO | Upward trend; stable growth | Regular annual seasonality | Mild short-term volatility |
| ED | Upward but slightly flatter trend | Clear yearly pattern | Noticeable fluctuations mid-2023–2024 |
| EXC | Sharp rise from 2020–2022, then steady | Strong yearly cycle | Higher volatility after 2022 |

Interpretation

The decomposition helps to understand the structure of time series data before modeling.

Stocks with strong trend and seasonality (like DUK and SO) are suitable for SARIMA/SARIMAX models,

while those with high cyclic volatility (like EXC) may benefit from LSTM models that can capture nonlinear dynamics.

– Show how to test Stationarity (Augmented Dickey-Fuller Test)

The Augmented Dickey-Fuller (ADF) test was applied to test stationarity.

All original price series were non-stationary ($p\text{-value} > 0.05$), so we applied differencing to make them stationary. ARIMA, SARIMA, and SARIMAX handle differencing automatically through parameters d and D

| Symbol | p-value Price |
|--------|---------------|
| AEP | 0.9 |
| DUK | 0.837 |
| SO | 0.928 |
| ED | 0.653 |
| EXC | 0.46 |

– Show how to create a Train/Test Split

| Forecast | train | test |
|-------------------|---------|---------------------------------|
| 7 day (1 week) | 6 month | last 5 trading days |
| 180 day (6 month) | 2 year | last 126 trading days |
| 365 day (1 year) | 5 year | last 256 trading days |
| LSTM (all period) | 10 year | last 5 / 126 / 256 trading days |

4. Compare at least three forecasting models and assess their performance.

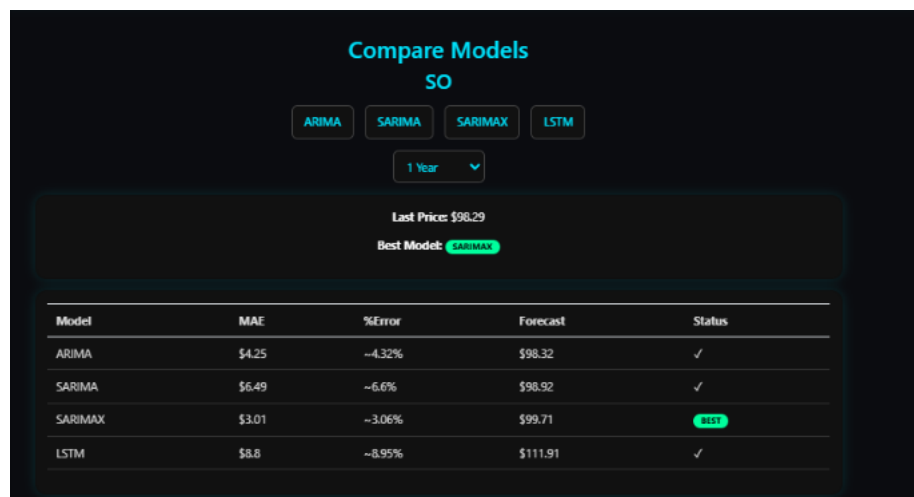
- Show how to choose the most suitable model.

All models were evaluated using Mean Absolute Error (MAE) from backtesting on the test data.

Smaller MAE indicates better forecasting accuracy.

$$MAE = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t|$$

In compare model page automatically retrieves stored MAE values from the stock_forecasts table and visualizes forecasts for each model.



User can select the best model, which has the lowest MAE. The UI will show “BEST” in the status.

The comparative performance analysis of the forecasting models is provided in Pages 20–22 of this report.

5. Analyze and summarize results

- The forecast results can help decide.

The forecast results generated by the DAG pipeline provide predicted stock prices for 1 week, 6 months, and 1 year ahead, using multiple models (ARIMA, SARIMA, SARIMAX, and LSTM).

MAE comparisons allow investors to identify the most reliable model for understanding potential trends, forecasted price ranges, and each model's relative confidence.

This information supports decision-making regarding stock buying, selling, and risk management, particularly for selected U.S. utility stocks: AEP, DUK, SO, ED, and EXC.

- What is the limitation of Forecasting in this dataset?

| Limitation | Explanation |
|------------------------|---|
| Market Noise | Stock prices can be influenced by unexpected macroeconomic events or sudden news, introducing unpredictability. |
| Limited Exogenous Data | Only oil, gas, and XLU are included. If these inputs are highly volatile, forecast accuracy may decrease. |
| Model Assumptions | ARIMA, SARIMA, and SARIMAX assume stationarity and linearity; sudden or extreme volatility can reduce forecast accuracy. |
| Data Granularity | ARIMA and SARIMA use only daily closing prices, potentially missing important external factors affecting the stock, while SARIMAX incorporates limited exogenous variables. |

6. Deployment

- Show how to use your forecasting model to apply to your project.

To apply our forecasting models in the project, we integrated them into an automated Airflow DAG pipeline. This pipeline retrieves market data, calculates technical indicators, and forecasts stock prices for a predefined list of US utility stocks (AEP, DUK, SO, ED, EXC) using multiple models — ARIMA, SARIMA, SARIMAX, and LSTM — across various horizons (7, 180, 365 business days). Both the forecast results and backtesting outcomes are stored in a PostgreSQL database. The pipeline is scheduled to run automatically at 19:30 (Bangkok time) on every business day. The deployment process consists of the following steps:

1. Data Retrieval and Preprocessing

- Historical market data and exogenous variables are pulled via a Spark job from external sources (Yahoo Finance) and stored in Parquet format.
- Technical indicators are computed automatically for each stock.

2. Backtesting and Forecasting

- For each stock, multiple models—ARIMA, SARIMA, SARIMAX, and LSTM—are applied across different forecasting horizons (7, 180, 365 business days).
- Backtesting is performed on historical data to calculate prediction errors (MAE), ensuring model reliability.
- Future forecasts are generated using the trained models, considering both historical trends and exogenous factors when applicable.

3. Automated Task Management

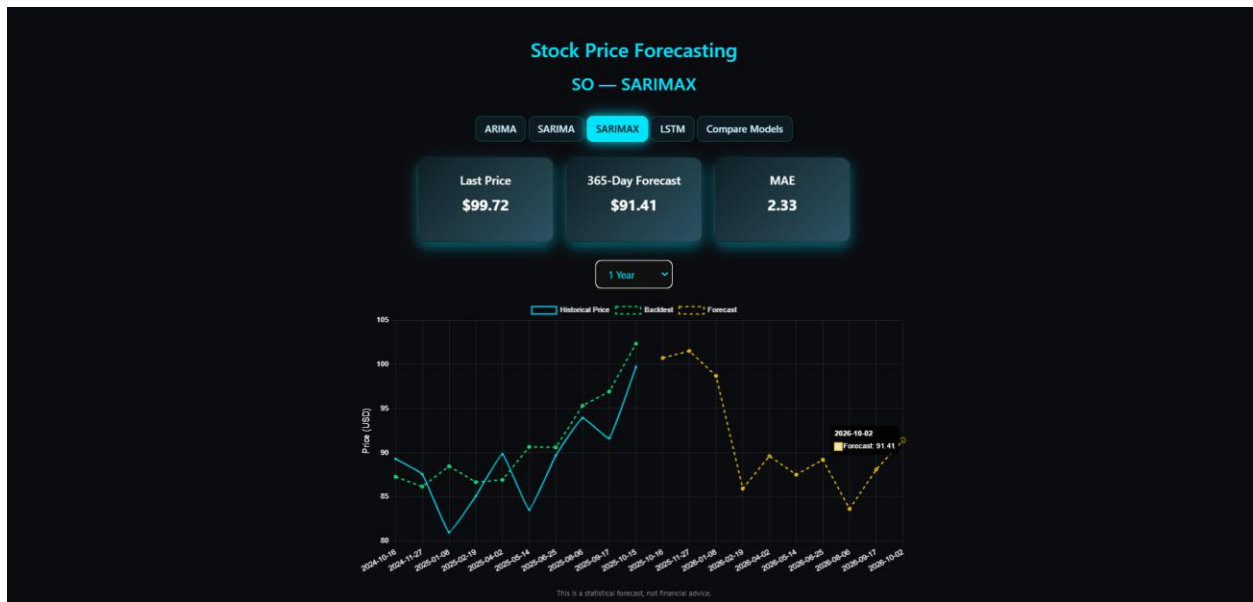
- The pipeline uses PythonOperators for model execution and TaskGroups to organize all models and horizons per stock.
- Only valid stocks with sufficient data are processed, while others are skipped.

4. Database Integration

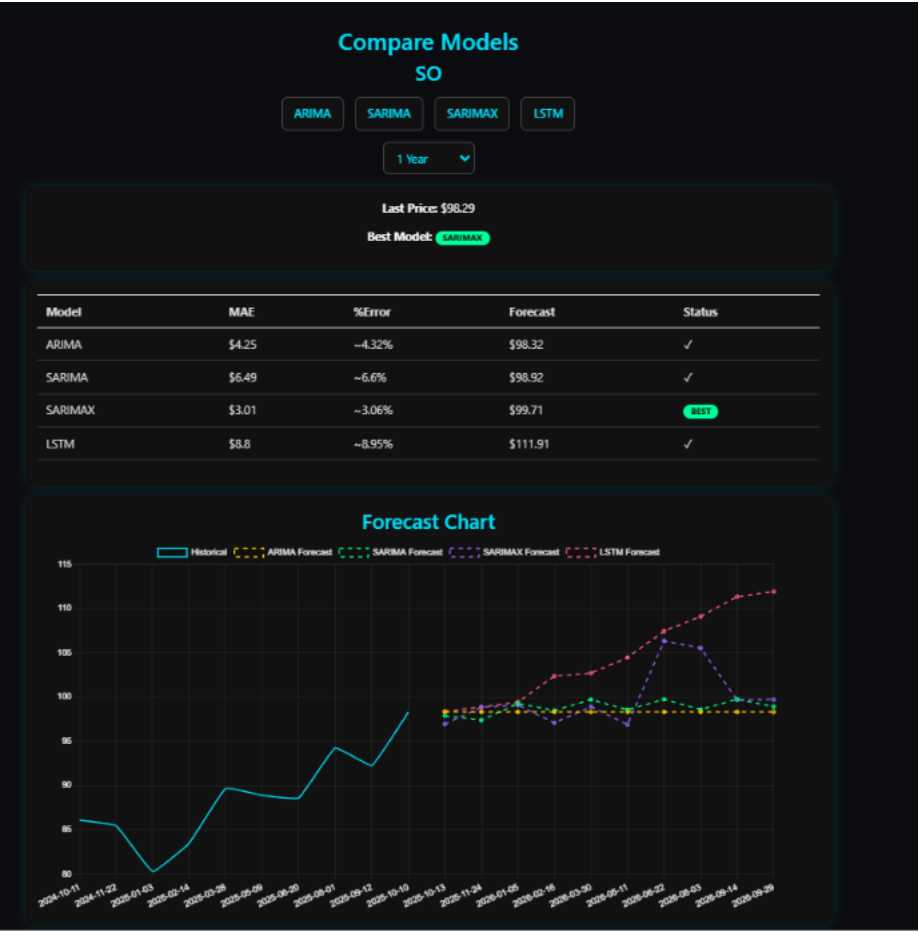
- Forecast and backtest results, along with metadata (last price, MAE), are stored in a PostgreSQL table (stock_forecasts) for downstream access.
- This enables real-time retrieval and visualization of forecasts through the web interface.

5. Web Deployment

- Forecasts are displayed in a Flask-based web application, allowing users to select a stock, choose a model, and view interactive charts of both backtesting results and future predictions.
- Users can also compare model performances across horizons, facilitating informed investment decisions.



Forecasting Page



Compare Models Page

7. Model Inputs and Parameters

- **Parameter:** Model Parameter Tuning (Grid Search) In the modeling part, we use grid search to find the best parameters for each model and each stock.

| Symbol | Model | Params | MAE |
|--------|---------|----------------------------|----------|
| AEP | SARIMA | ((0, 1, 0), (1, 1, 0, 20)) | 3.876164 |
| AEP | SARIMAX | None | inf |
| DUK | ARIMA | (1, 1, 1) | 5.232380 |
| DUK | SARIMA | ((2, 1, 2), (1, 1, 1, 63)) | 3.467572 |
| DUK | SARIMAX | None | inf |
| SO | ARIMA | (0, 1, 1) | 3.506928 |
| SO | SARIMA | ((1, 0, 0), (1, 1, 0, 20)) | 3.285717 |

| Symbol | Model | Params | MAE |
|--------|---------|----------------------------|----------|
| SO | SARIMAX | None | inf |
| ED | ARIMA | (2, 1, 2) | 4.842780 |
| ED | SARIMA | ((2, 0, 2), (1, 1, 0, 63)) | 4.580979 |
| ED | SARIMAX | None | inf |
| EXC | ARIMA | (2, 1, 2) | 3.454461 |
| EXC | SARIMA | ((1, 1, 1), (1, 1, 0, 20)) | 1.707662 |
| EXC | SARIMAX | None | inf |

| Symbol | Model | Params | MAE |
|--------|---------|----------------------------|----------|
| AEP | SARIMAX | ((2, 0, 2), (0, 0, 1, 20)) | 3.211657 |
| DUK | SARIMAX | ((0, 1, 2), (1, 1, 1, 63)) | 2.428156 |
| SO | SARIMAX | ((0, 1, 2), (1, 0, 1, 20)) | 1.851084 |
| ED | SARIMAX | ((2, 1, 1), (1, 1, 1, 63)) | 5.184310 |
| EXC | SARIMAX | ((1, 0, 0), (0, 1, 1, 63)) | 1.754063 |

- **Exogenous Variables:** For the SARIMAX model, we include exogenous variables such as Crude Oil (CL=F), Natural Gas (NG=F), and the Utilities Sector ETF (XLU) to enhance forecasting accuracy.

These variables are correlated with utility stock prices because changes in energy prices directly affect production costs and investor sentiment in the utilities sector.

Before using them in the model, each exogenous variable was forecasted separately to extend its future values.

Since future data for oil, gas, and XLU are unknown, we applied a Random Walk Simulation based on their historical daily returns:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}, \quad P_{t+1} = P_t(1 + \epsilon_t)$$

where ϵ_t represents a random shock term drawn from a normal distribution whose mean (μ) and standard deviation (σ) are derived from the historical return series.

This method assumes that future movements of exogenous variables follow similar volatility and drift as their historical behavior.

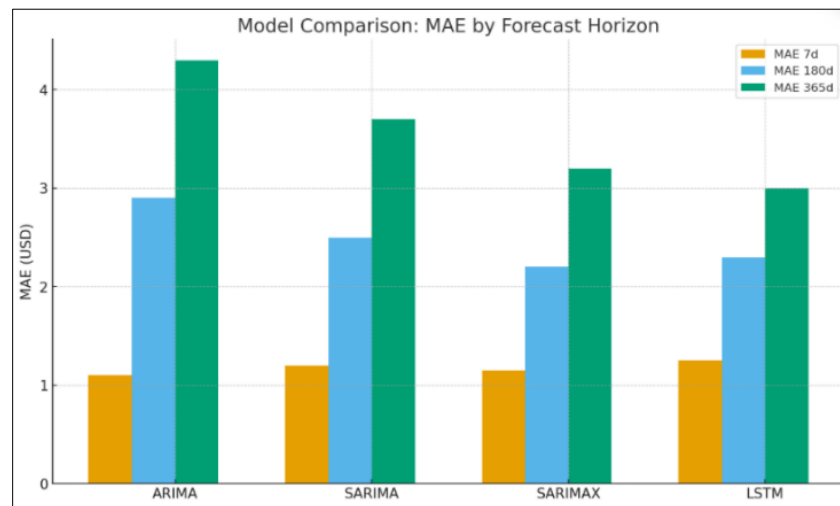
The simulated exogenous series are then fed into the SARIMAX model to generate more realistic stock price forecasts.

Part B: AI Model Development and Performance Evaluation

1. Your tools or models in AI-model development, and why do you choose them for the project?

a. Show some comparison among the alternatives.

Model Comparison



This chart compares the Mean Absolute Error (MAE) of our four models

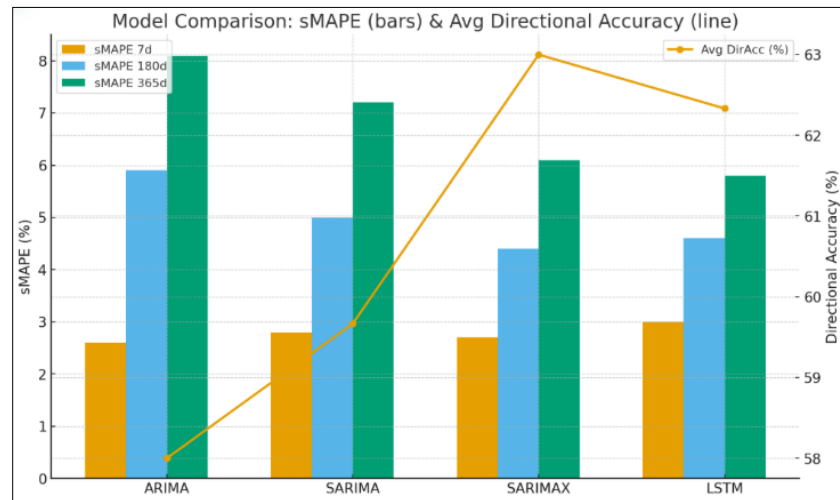
ARIMA, SARIMA, SARIMAX, and LSTM across three forecast horizons: 7, 180, and 365 days.

We can see that all models perform well on short-term forecasts (7 days), with MAE around 1 to 3 USD, indicating good short-term precision.

As the forecast horizon extends to 180 and 365 days, errors naturally increase for all models due to uncertainty accumulation.

However, SARIMAX consistently delivers the lowest MAE, especially for medium and long horizons, because it integrates exogenous variables like oil, gas, and XLU — which help capture broader market drivers.

The LSTM model performs competitively, especially for longer horizons, suggesting it can learn underlying non-linear patterns, but it sometimes over-smooths short-term fluctuations in stable sectors like utilities.



This chart compares the models using two complementary metrics

sMAPE (%) represented by the colored bars measures the relative prediction error in percentage terms.

Directional Accuracy (%) shown as the orange line indicates how often the model correctly predicts the direction of movement (up or down).

From the results

- SARIMAX achieves the lowest sMAPE and the highest Directional Accuracy (~63%), showing that it performs best in both accuracy and trend prediction.
- LSTM follows closely, maintaining solid trend detection even with slightly higher sMAPE, reflecting its strength in capturing non-linear patterns.
- SARIMA performs consistently well across different horizons, while ARIMA serves as a simple but reliable baseline model.

Overall, the combination of low sMAPE and high Directional Accuracy indicates that SARIMAX provides the most balanced and dependable forecasting performance among the models.

Models evaluated (for 7d, 180d, 365d horizons)

| | |
|---------|--|
| ARIMA | fast, strong on short-term autocorrelation; good short-horizon baseline. |
| SARIMA | adds seasonality (weekly/monthly patterns), often more stable than ARIMA for short - mid horizon. |
| SARIMAX | SARIMA + exogenous drivers (Crude oil: CL=F, Natural gas: NG=F, Utilities ETF: XLU); typically best all horizon. |
| BiLSTM | neural sequence model using Close price plus technical indicators (RSI, EMA10/50, MACD, Bollinger %b, 20-day volatility) + exogenous LSTM model performs especially for on mid-long horizon. |

Selection principle: choose the lowest MAE, then sanity check the forecast cur

Tool Comparison

| Feature / Criteria | Matplotlib | MLflow | TensorBoard |
|-----------------------------------|---|---|---|
| Purpose | Static visualization for charts and metrics | Experiment tracking, model versioning | Real-time monitoring and visualization of deep learning models |
| Integration | Generic (works with any data) | Integrates with Python / ML pipelines | Tight integration with TensorFlow and Keras |
| Real-time Tracking | Not supported | Limited (metrics after run) | Fully supported (live updates during training) |
| Multi-run Comparison | Manual plotting required | Supported via run history | Native overlay comparison of multiple experiments |
| Ease of Use | Simple, flexible plotting but manual | Requires setup and configuration | Plug-and-play with TensorFlow logs (--logdir) |
| Key Use in Project | Visualize model metrics (MAE, RMSE, sMAPE, DirAcc%) and final comparisons | Track experiment versions (not real-time) | Monitor LSTM training loss, validation loss, and learning rate schedules live |
| Best For | Reporting & presentation figures | Model lifecycle management | Model diagnostics, convergence monitoring, and EarlyStopping tuning |
| Limitation | No real-time feedback; manual plot updates | Weak visualization; not focused on deep learning | Limited to TensorFlow-based workflows |
| Why Chosen / Role in This Project | Used to summarize and present final metric results | Reviewed but not adopted due to lack of visualization | Main monitoring tool for LSTM; shows convergence behavior and optimal epoch selection |

b. Demonstrate how you tailor the tool to your specific project.

TensorBoard was specifically tailored for the forecasting component by focusing on LSTM-relevant metrics — particularly training loss, validation loss, and learning rate schedules — which are the most critical for monitoring convergence in time-series models.

Instead of plotting every possible metric, the setup was simplified to highlight only the information necessary to efficiently identify overfitting, model performance, and the optimal number of epochs.

Through these customizations, TensorBoard becomes a suitable diagnostic tool, helping to determine whether the model has converged, whether its performance is sufficient, whether learning is stable, and when to automatically stop training using EarlyStopping.

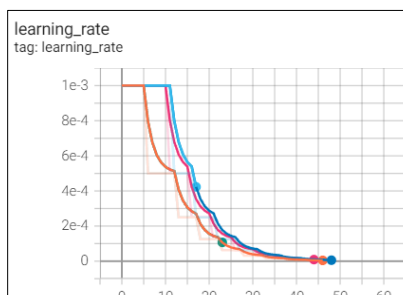
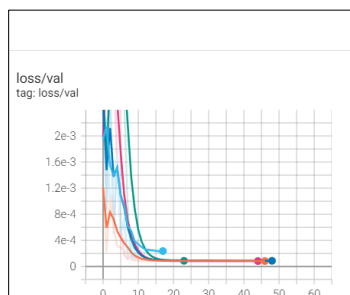
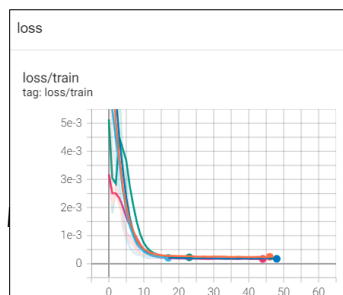
This tailoring directly supports model tuning and improves training efficiency within the test code.

C. Can your model be tuned to the new incoming data?

Yes, our models can be tuned to new incoming data. Within the Airflow pipeline, an automated system retrieves market data, calculates technical indicators, and forecasts stock prices for a predefined set of utility stocks, including AEP, DUK, SO, ED, and EXC. The system employs multiple forecasting models — ARIMA, SARIMA, SARIMAX, and LSTM — across different forecasting horizons (7, 180, and 365 business days). Both the forecast results and backtesting outcomes are stored in a PostgreSQL database. The system is scheduled to run automatically at 19:30 (Bangkok time) on every business day.

2. Show the performance assessment of model

| ticker | steps | MAE\$ | RMSE\$ | sMAPE% | MAPE% | MAE_ret | DirAcc% | MAE_norm% | Infer_ms_future | Infer_ms_future_per_step |
|--------|-------|--------------------|--------------------|--------------------|--------------------|-----------------------|---------|--------------------|-------------------|--------------------------|
| AEP | 7 | 3.4065902709960936 | 3.782464955263111 | 2.9557584036838613 | 2.9030818146646133 | 0.012009414070063503 | 25.0 | 3.002459279203402 | 43734.72959999344 | 8745.945919998689 |
| DUK | 7 | 2.23206787109375 | 2.294943725497134 | 1.805251088781178 | 1.7882167808113718 | 0.005666235481010633 | 50.0 | 1.8360351044774827 | 37153.06340000825 | 7430.61268000165 |
| SO | 7 | 1.5361251831054688 | 1.634787000392614 | 1.6148341926674796 | 1.600230358866275 | 0.00558196842211965 | 50.0 | 1.6360903109252958 | 43341.20259998599 | 8668.240519997198 |
| ED | 7 | 1.660028076171875 | 1.7410376209181098 | 1.6717660617100776 | 1.6565836135806213 | 0.0038725538275203547 | 75.0 | 1.7001516656890159 | 56682.37809999846 | 11336.475619999692 |
| EXC | 7 | 1.1915458679199218 | 1.3173712700375888 | 2.6034398031132158 | 2.5628311297686697 | 0.008813742416715247 | 75.0 | 2.6496460647334796 | 59895.62870000373 | 11979.125740000745 |



| | |
|--------------------------|---|
| MAE (USD) | absolute price error in dollars directly answers “How much money is the mistake?” |
| RMSE (USD) | penalizes large errors useful to control occasional big misses. |
| sMAPE (%) / MAPE(%) | scale-free error for cross-ticker and cross-period comparison. |
| MAE_ret (daily return) | error on daily returns measures how well we capture day-to-day changes. |
| Direction Accuracy (%) | up/down correctness — critical for directional strategies, hedging, and signal safety. |
| MAE_norm (%) | MAE normalized by the latest price — intuitive “% of price” error. |
| Infer_ms_future | Total time (latency) in milliseconds to perform the future forecast for the full 7 days |
| Infer_ms_future_per_step | Average time (latency) in milliseconds to perform the future forecast per day |

a. Inference time measurement.

The LSTM inference time depends on the forecasting horizon. To allow fair comparison across 7-day, 180-day, and 365-day horizons, we report the per-step inference time (`Infer_ms_future_per_step`).

The average time to forecast a single trading day ranges from 7,430 ms (DUK) to 11,979 ms (EXC):

- DUK: 7,430 ms
- SO: 8,668 ms
- AEP: 8,747 ms
- ED: 11,336 ms
- EXC: 11,979 ms

Forecasting is performed within the Airflow pipeline. The web interface pages (forecasting page and compare models page) simply retrieve predictions from the database, so the display time on the web is minimal.

b. What are your evaluation metrics to assess the model performance and how do they relate to the project?

MAE (Mean Absolute Error)

- Measures the average magnitude of forecast errors in USD, without considering direction.
- Indicates how close the model's predictions are to the true stock prices.
- Lower MAE values mean higher accuracy.

RMSE (Root Mean Squared Error)

- Penalizes larger errors more heavily than MAE.
- Useful for detecting models that produce occasional large deviations.
- Helps assess overall stability of predictions.

sMAPE (Symmetric Mean Absolute Percentage Error)

- Expresses error as a percentage of the actual value, making it comparable across tickers with different price ranges.
- This metric is ideal for cross-stock comparison among AEP, DUK, SO, ED, and EXC.

Directional Accuracy (DirAcc%)

- Measures how often the model predicts the correct direction (up or down) of price movement.
- Especially important for trading-oriented or trend-following decisions, where direction matters more than exact price.

BiLSTM Model

Model Structure:

The forecasting model uses a Bidirectional LSTM (BiLSTM) architecture.

| Layer | Type | Parameters / Notes |
|-------|--------------------|--|
| 1 | Bidirectional LSTM | 64 units, <code>return_sequences=True</code> |
| 2 | Dropout | 0.2 (regularization to prevent overfitting) |
| 3 | Bidirectional LSTM | 32 units |
| 4 | Dense | Output = 1 (recursive) or K (direct mode) |
| - | Loss | Huber (robust to outliers) |
| - | Optimizer | Adam |
| - | Callbacks | EarlyStopping, ReduceLROnPlateau |

Target Definition:

- The main prediction target is daily return, computed as
$$\text{ret}_t = \text{Close}_t / \text{Close}_{(t-1)} - 1$$
- In direct mode, the model predicts block-summed log returns using $\log_{1p}(\text{ret})$ to ensure numerical stability and additive consistency for long-term horizons.

Input Feature Engineering:

(a) Core Features

- `target_ret` (daily return of closing price)
- Always placed in column 0 for inverse scaling consistency.

(b) Technical Indicators

Derived from the closing price:

- **EMA (10, 50)** – trend detection
- **RSI (14)** – momentum and overbought/oversold levels
- **MACD, MACD Signal** – crossover-based trend strength
- **Bollinger %B** – normalized position within Bollinger Bands
- **Rolling Volatility (20-day)** – local market uncertainty

(c) Exogenous Drivers

External features are integrated as exogenous variables

- **CL=F (Crude Oil)**
- **NG=F (Natural Gas)**
- **XLU (Utilities ETF)**

Each is transformed into daily returns, aligned by business days, and standardized.

If future exogenous data is unavailable, a random-walk simulation is generated based on past return statistics (μ , σ).

Data Scaling:

- A single `StandardScaler` is applied across all features (including `target_ret`).
- Scaling ensures consistent magnitudes and enables inverse transformations for return-based predictions.

Sequence Windowing:

The input data is segmented into rolling time windows:

- **Lookback:** 120–200 days (tuned based on forecast horizon).
- **Shapes:**
 - X: (num_samples, lookback, num_features)
 - y: (num_samples, 1) or (num_samples, K)
- Splits use a **chronological 80/20 train-validation** split — no shuffling to preserve temporal integrity.

Forecasting Horizon Modes:

- **Recursive mode ($H \leq 63$ business days)**

Predict one day at a time; feed the forecasted return back into the next window.

Suitable for short-term forecasts.
- **Direct mode ($H > 63$ business days)**

Predict multiple days (e.g., 5-day blocks) and distribute results daily.

Reduces cumulative error for long-term forecasts.

Forecast Post-Processing:

- **Drift blending:** Smooths long-term forecasts by blending with historical drift trends.
- **Return clipping:** Limits extreme daily returns (e.g., $\pm 7\text{--}8\%$) to prevent unrealistic fluctuations.
- **Business-day mapping:** Aligns predictions with actual trading days, ensuring consistency in the horizon length.

Training Optimization:

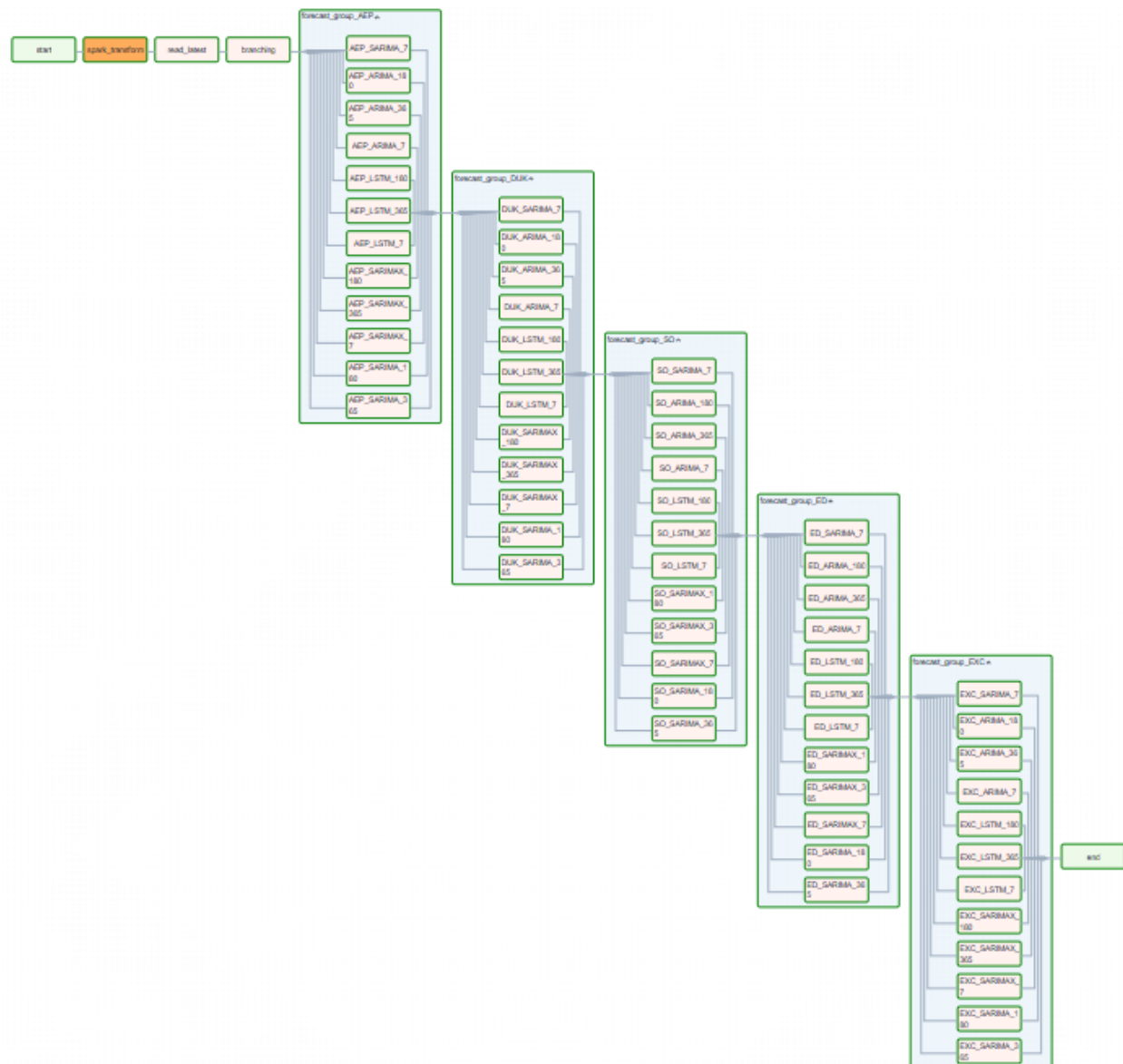
- **Regularization:** Dropout = 0.2
- **EarlyStopping:** patience \approx 10 epochs
- **ReduceLROnPlateau:** adaptive learning-rate tuning
- **Deterministic mode:** fixed random seeds and TF_DETERMINISTIC_OPS=1 for reproducibility

Evaluation Metrics:

- MAE, RMSE, sMAPE, MAPE – price-level accuracy
- MAE_ret, DirAcc% – return and directional accuracy
- **Inference timing:** measured as Infer_ms_future_per_step for performance evaluation.

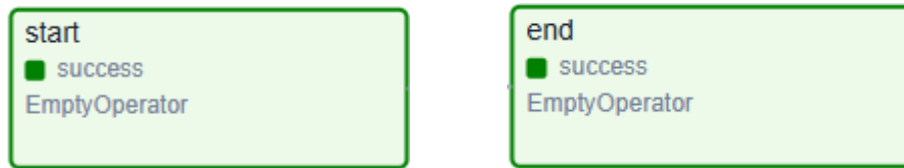
TensorBoard Monitoring: Tracks training loss, validation loss, and learning rate.

Part C: Airflow DAG Concepts



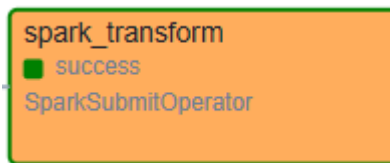
This pipeline is an automated system that retrieves market data, calculates technical indicators, and forecasts stock prices for a predefined set of utility stocks (AEP, DUK, SO, ED, EXC). It employs multiple forecasting models — ARIMA, SARIMA, SARIMAX, and LSTM — across various horizons (7, 180, 365 business days). Finally, both the forecast results and backtesting outcomes are stored in a PostgreSQL database. The system is scheduled to run automatically at a specific time — in this case, 19:30 (Bangkok time) on every business day.

1. **Start/Stop** – Use DummyOperator to define the boundaries of your DAG.



DummyOperator creates tasks that perform no work and act as markers for the start and end of the DAG. In this pipeline start marks the beginning and end marks the finish. Using these markers clarifies the workflow layout

2. **PySpark Processing** – Use SparkSubmitOperator for data processing.

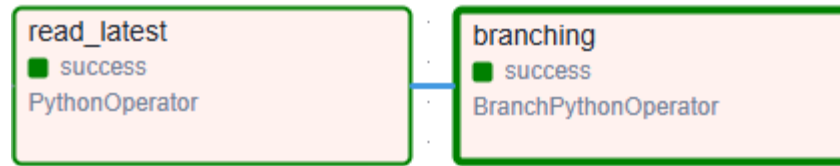


Spark job (key behavior in src\spark\applications\spark_jobs\pull_yf.py):

The script downloads price series and exogenous series from yfinance, builds a unified DataFrame and writes market.parquet and exog.parquet into /opt/airflow/src/spark/data/spark_out/. These Parquet files become the upstream data source.

SparkSubmitOperator is used to execute the PySpark script that extracts and prepares raw market/exogenous data and stores it as Parquet. This decouples heavy data ingestion/ETL from the model tasks and produces a reproducible data artifact consumed by the forecasting steps.

3. XCom – Share results between tasks via XCom.

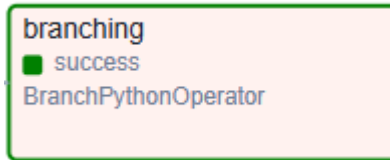


| △ Details Graph Gantt <> Code Audit Log Logs XCom Task Duration | |
|---|---|
| Key | Value |
| last_close_list | [{"symbol": "AEP", "last_close": 116.91000366210938}, {"symbol": "DUK", "last_close": 124.70999908447266}, {"symbol": "SO", "last_close": 96.12999725341797}, {"symbol": "ED", "last_close": 100.80000305175781}, {"symbol": "EXC", "last_close": 46.65999984741211}] |

The `read_latest` task reads the latest stock prices from a Parquet file and pushes the results (last closing prices of each stock) to XCom using the key "last_close_list". The `decide_branch` task then pulls this data from XCom to determine which forecast tasks should run based on a defined threshold.

This approach allows tasks to share data dynamically without directly passing arguments, enabling downstream tasks to make decisions based on the results of upstream tasks.

4. **Branching/Decision** – Use the BranchPythonOperator to control the workflow based on specific conditions.

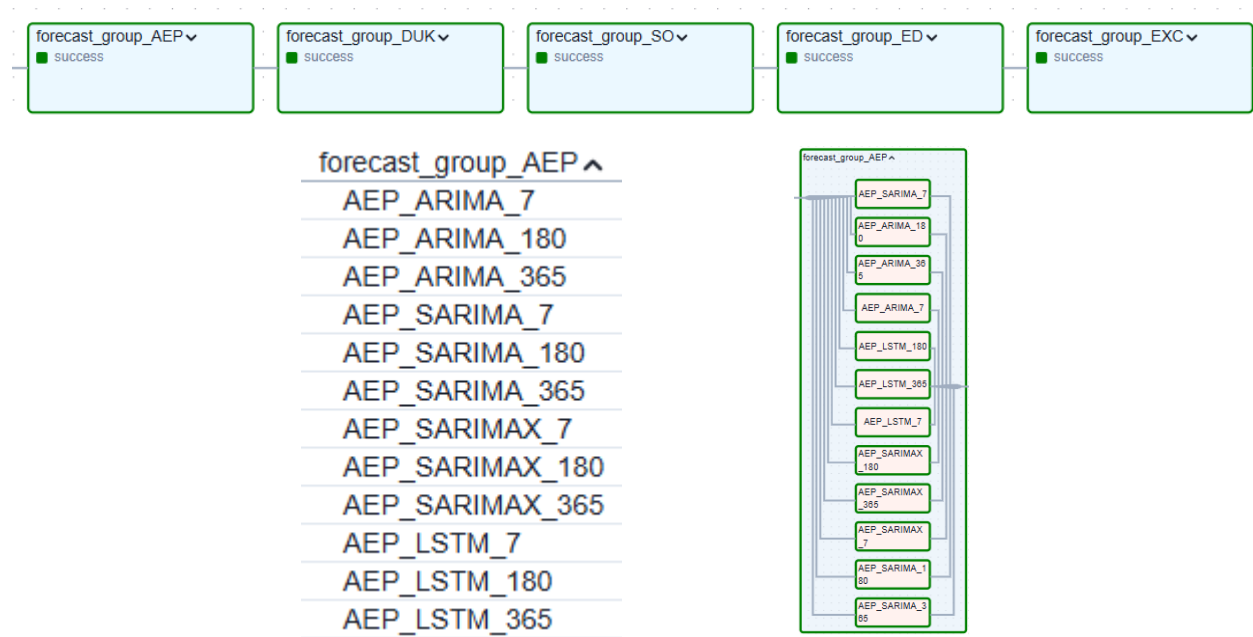


The `decide_branch` function implements conditional branching in the Airflow DAG using the BranchPythonOperator. It reads the last closing prices of stocks from XCom (provided by the `read_latest` task), compares each price against a defined threshold, and generates a list of task IDs corresponding to the forecast tasks that should run.

- If the last closing price of a stock exceeds the threshold, all forecast tasks for that stock (for each model and horizon) are added to the selected list.
- If no stock meets the condition, the function returns "end", a placeholder task connected to the downstream end task, effectively terminating the branch while allowing the workflow to continue in a controlled manner.
- The function returns a list of task IDs that Airflow will execute next, dynamically controlling the workflow based on the latest stock data from XCom.

Note: The threshold is intentionally set very low because the main purpose is only to filter out any missing or None values in the data, not to perform meaningful price-based filtering.

Forecast Tasks



We utilize TaskGroup to manage and organize the modeling phase

- `forecast_group_{symbol}` (TaskGroup): Groups all forecast tasks for a single stock symbol (AEP, DUK, etc.), covering all models (ARIMA, SARIMA, SARIMAX, LSTM) and horizons (7, 180, 365 business days). This structure streamlines visualization and management in the Airflow UI.
 - o Core Action: The underlying `run_model` function executes the model training, backtesting, and forecasting. Crucially, it then stores the results (forecast data, backtest data, and MAE) into the PostgreSQL table `stock_forecasts` using an UPSERT mechanism, ensuring critical outcomes are persistently saved.
- Sequential Dependencies: The `prev_group` variable maintains sequential dependencies (`prev_group >> forecast_group`), ensuring that all modeling tasks for one stock complete before the next stock's tasks begin, creating an orderly execution flow.

Part D: User Interface Design and Component Overview

1. User details

The primary users of this web application are individual investors, financial analysts, and portfolio managers who are interested in the US Utilities Sector. These users typically seek stable, long-term investments with moderate risk and consistent dividend yields.

The selected stocks—AEP, DUK, SO, ED, and EXC—are well-established utility companies with reliable cash flows and essential services, making them attractive to risk-averse investors who prioritize steady returns over high volatility. Users of this platform are likely to value:

- In-depth stock analysis: Access to historical price trends, key financial indicators, and interactive visualizations allows them to monitor performance closely.
- Data-driven forecasting: Advanced models (ARIMA, SARIMA, SARIMAX, LSTM) to predict future stock prices and backtesting provide actionable insights on potential price movements.
- Efficient decision-making: Automated updates and model comparisons help users quickly identify the most reliable forecasting method.
- Market news integration: Access to the latest relevant news articles ensures users stay informed about sector developments that may impact their investments.

2. Overview of UI design, techniques, and libraries

UI design

Happy Stock Web Application is designed with a focus on clarity, interactivity, and data visualization. The goal is to provide investors and analysts with a seamless experience for exploring, comparing, and forecasting stock prices in the US utilities sector.

The overall layout is implemented through a base template (`base.html`) using Flask's Jinja2 templating engine, ensuring visual consistency across all pages. The design employs a glassmorphism-style navigation bar, intuitive card layouts, and responsive chart sections that adapt to different screen sizes.

Color schemes emphasize dark backgrounds with neon-accented tones to improve readability of financial charts and align with a modern analytical aesthetic. Interactive elements such as dropdown filters, toggles, and active-state buttons are employed to enhance navigation and engagement. Typography and spacing follow Bootstrap 4 standards, ensuring accessibility and readability across devices.

Techniques

The web application uses a modular and data-driven approach, integrating dynamic rendering, responsive layouts, and real-time visualization. Key techniques include:

1. **Template Inheritance (Jinja2):**

A hierarchical template structure is implemented using `base.html` as the foundation, from which other pages extend. This ensures consistent layout elements such as the navigation bar, alert messages, and overall page structure. The base template also links to a primary stylesheet (`style.css`), which serves as the main theme for the web application, maintaining consistent colors, typography, and visual effects across all pages.

2. **Dynamic Data Binding:**

Server-side Python functions pass dynamic data into the HTML via Jinja2 expressions. These variables are then transformed into JavaScript-friendly formats using `|tojson`, enabling real-time data visualization without reloading the page.

3. **Client-Side Interactivity:**

JavaScript and external libraries (`Chart.js` and `Plotly.js`) handle user-driven updates such as changing forecast horizons, and switching models. Interactive charts automatically update when users select different options, providing immediate visual feedback.

4. **Responsive Design and Accessibility:**

Using Bootstrap's grid system and utility classes, the interface is fully responsive and automatically adjusts to fit various screen sizes and resolutions. Semantic HTML elements are employed to enhance accessibility and maintain a clear structure. Separate CSS files maintain modular styling, improving clarity and ensuring theme consistency across different pages.

5. Data Visualization Techniques:

Provide users with a comprehensive and interactive view of stock data, using visualizations such as line charts that distinguish between actual, backtested, and predicted values, bar charts showing percentage changes over specific time periods with color-coding for gains and losses, and heatmaps that reveal the correlations between the prices of all five stocks.

Interactive features are included in all visualizations, allowing users to hover over data points to view exact values and context such as forecast horizon and forecasted value. Users can filter by time range or model, dynamically updating charts to focus on relevant data. Tooltips, legends, and labels enhance clarity, while smooth transitions and responsive layouts ensure readability across devices.

Libraries

| Category | Library / Tool | Purpose |
|----------------------------|--------------------|---|
| Frontend Framework | Bootstrap 4 | Provides responsive layout, consistent spacing, and built-in components such as navbar, buttons, and forms. |
| Icons | Font Awesome 6 | Supplies professional icons (e.g., chart-line icon in the navbar) for improved UI recognition. |
| Charting Library | Chart.js | Used to create interactive charts for stock analysis, model forecasting, and performance comparison. |
| Interactive Visualization | Plotly.js | Enables dynamic and responsive visualizations throughout the web application. |
| Interactive Trading Charts | TradingView Widget | Provides real-time interactive candlestick charts for detailed stock analysis. |
| Templating Engine | Jinja2 (Flask) | Enables HTML template inheritance, dynamic variable rendering, and integration with backend data pipelines. |
| Utility Scripts | jQuery & Popper.js | Support Bootstrap's interactive components such as dropdowns, alerts, and navigation toggles. |

3. Detailed explanation of each component

Stock Heatmap Page



Template Inheritance: The page extends base.html, reusing the navbar, flash messages, and global styles for consistency across the site.

Page Heading: A prominent `<h2>` heading displays "Stock Heatmap" in a bright blue color, matching the dark background theme.

Legend: Shows the performance range from -3% to +3% using a color gradient bar. Helps users understand stock returns visually

Stock Grid:

- Stocks are displayed as clickable cards (stock-card) in a responsive grid-container.
- Each card links to the stock's detail page, with background color reflecting its performance.

Stock Card Components

- **Symbol:** Displays the stock ticker.
- **Price:** Shows the current price.
- **Change:** Shows percentage change with color indication (green for positive, red for negative).
- **Hover Effect:** Cards scale up and glow on hover, with a tooltip showing the symbol.
- **Data Update:** Automatically fetch stock data every Monday to Friday from 9:30 AM to 4:00 PM NY time and update every 1 minute.

Last Updated Info: Displays the last updated timestamp, or "N/A" if unavailable, with styling for visibility and hover interaction.

Animations & Visual Effects:

- Background has a subtle animated particle grid.
- Positive and negative cards feature pulsing glow animations.
- Enhances interactivity and visual appeal.

Stock's Detail Page



Template Inheritance: The page extends base.html, allowing it to use the common layout, navbar, flash messages, and global styles without code duplication.

Page Heading: Displays the stock ticker (stock.symbol) as a large heading in blue, matching the dark-themed background.

Navigation Links:

- Provides News, Analytics, and Forecasting buttons for quick access to different information about the stock.
- Back-link buttons include hover effects to enhance UX and interactivity.

TradingView Chart:

- Embeds an interactive stock chart via TradingView widget.
- Responsive and dark-themed to match the page design.

Stock Info Box:

- Shows key data such as current price and percentage change.
- Uses a dark background with shadow for readability and emphasis.

Background & Visual Effects

- Animated grid-style background adds a dynamic sci-fi feel.
- Chart and info boxes use shadow and rounded corners for aesthetic appeal.

News Page



Template Inheritance: Inherits from base.html to reuse the layout, navbar, flash messages, and global styles without duplicating code.

Page Heading: Displays the main heading “News” and the stock ticker (symbol) linked back to the Stock Detail page, styled with blue color and large font for prominence.

News List:

- Shows stock-related news as a vertical list with spacing between items for readability.
- Each news item is clickable and opens in a new tab for full article access.

News Item Components

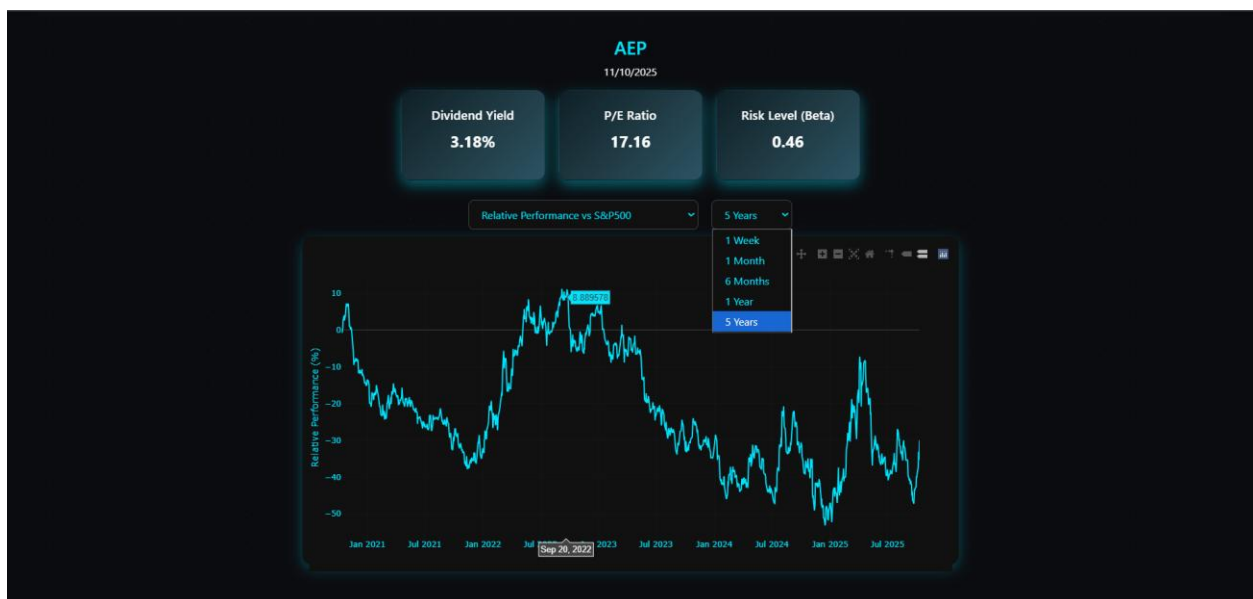
- **Headline:** Clickable news text linking to the full article.
- **Date:** Displays the publication date for context.
- **Hover Effect:** Each card slightly lifts and glows when hovered to enhance interactivity.

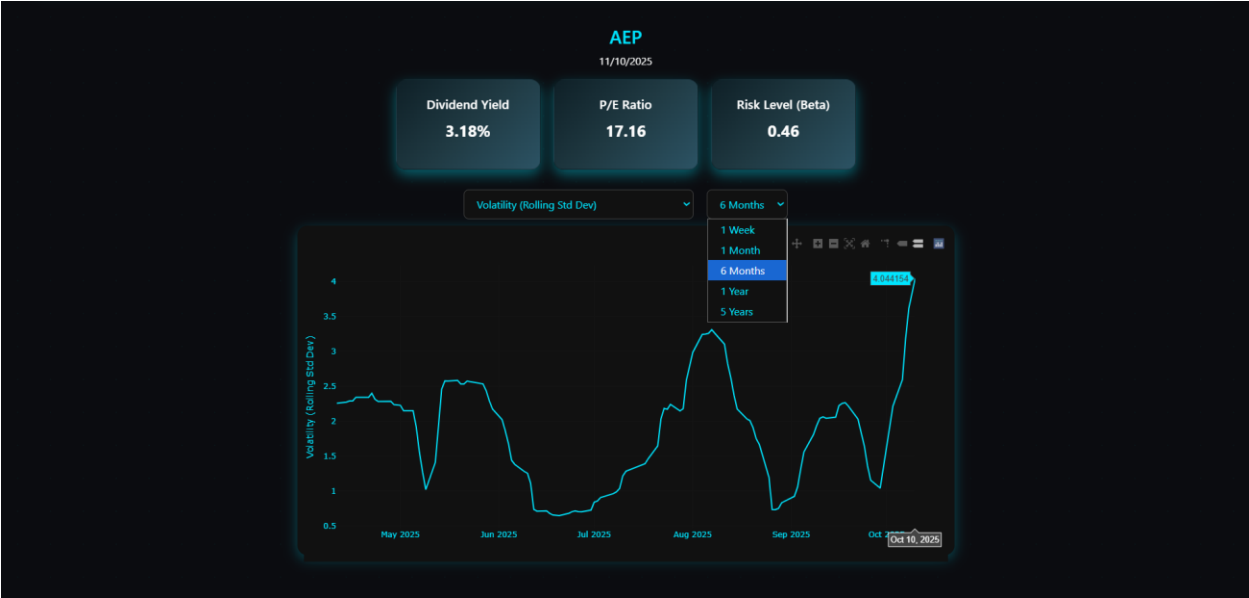
- **Data Update:** Automatically fetch news data every Monday to Friday at 7 PM Bangkok time.

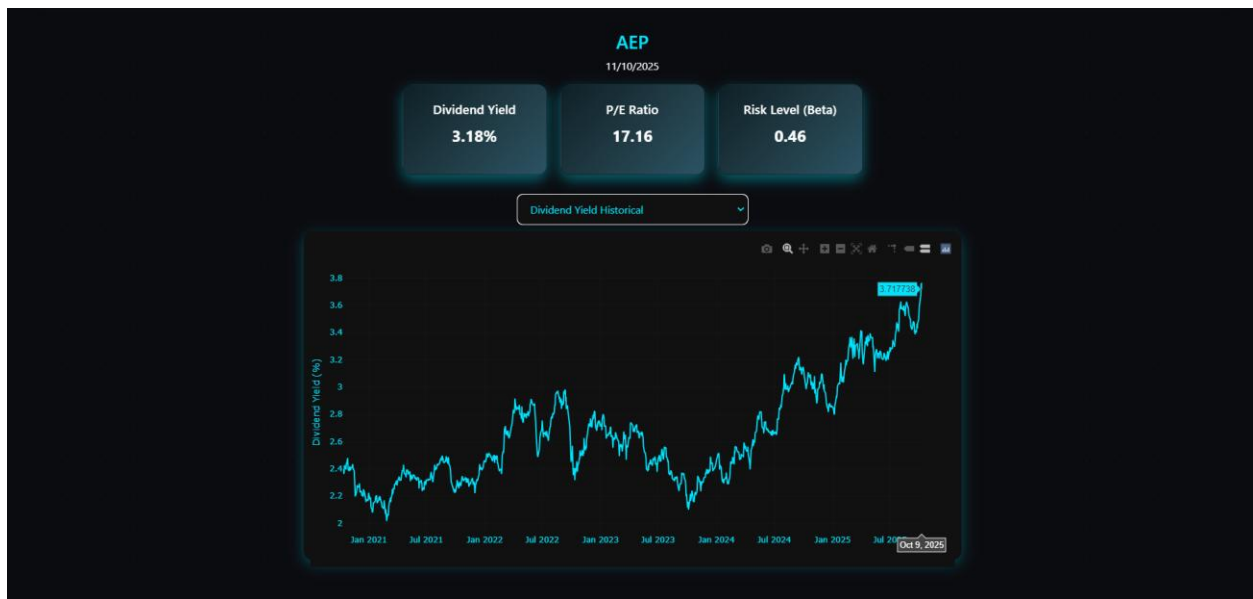
Background & Visual Effects

- The page background includes a subtle animated grid pattern consistent with the dark sci-fi theme.
- News cards feature a dark background, shadow, and blue border accent for visual emphasis and structured layout.

Stock Analytics Page







Template Inheritance: Inherits from base.html to reuse the layout, navbar, flash messages, and global styles without duplicating code.

Page Heading: Displays the stock symbol as a heading and links back to the Stock Detail page for easy navigation.

Last Updated: Shows the most recent update date for the stock data to inform users about its timeliness.

Overview Cards:

- Highlights key stock metrics such as Dividend Yield, P/E Ratio, and Beta (Risk Level).
- Uses gradient backgrounds, shadows, and hover effects to enhance visual appeal and interactivity.

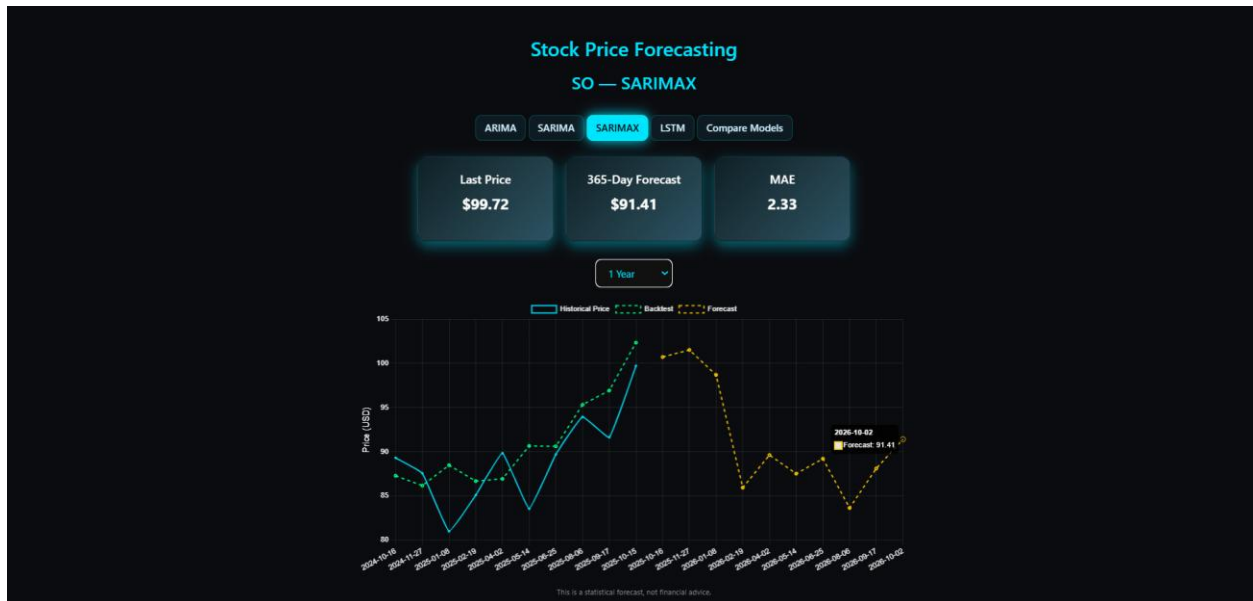
Chart Controls:

- Dropdown menus allow users to select the chart type and time range (1 week to 5 years).
- Certain charts automatically hide the time filter when not applicable.

Chart Section:

- Interactive charts are displayed using Plotly.js, with each chart occupying a responsive container.
- Available charts include:
 - Price vs MA50/MA200
 - Relative Performance vs S&P500
 - Volatility (Rolling Std Dev)
 - Revenue & Net Income Trend
 - Dividend Yield Historical
- Charts use color, shadows, and dark theme to maintain a consistent sci-fi style.

Forecasting Page



Template Inheritance: Inherits from base.html to reuse the layout, navbar, flash messages, and global styles without duplicating code.

Page Heading: Displays the page title “Stock Price Forecasting” along with the selected stock symbol and forecasting model, including a link back to the Stock Detail page for easy navigation.

Model Switch:

- Buttons to select forecasting models (ARIMA, SARIMA, SARIMAX, LSTM) and a link to compare models.
- Highlights the active model for easy navigation and clarity.

Overview Cards:

- Display key stock and forecast metrics such as Last Price, Forecasted Price, and MAE (Mean Absolute Error).
- Cards use gradient backgrounds, shadows, and hover effects to emphasize important information.

Forecasting Filters:

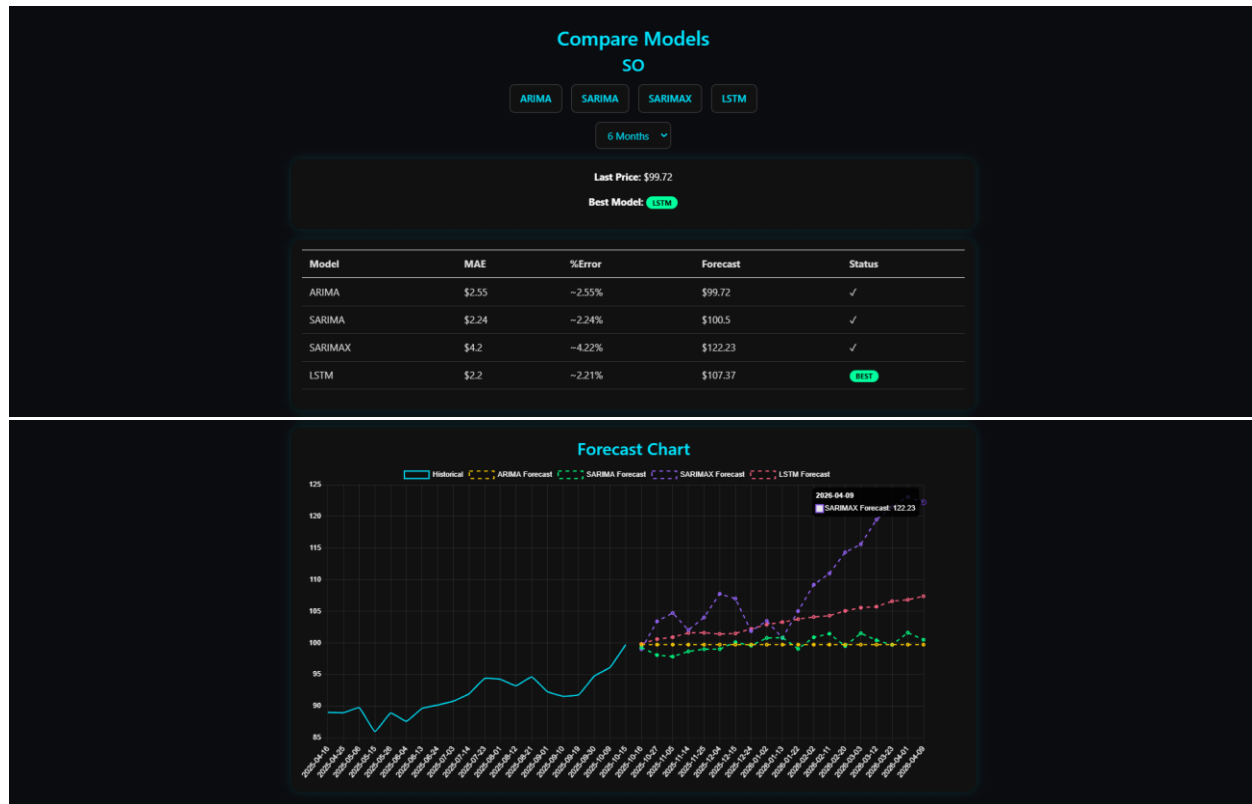
- Dropdown menu to select the forecast horizon (1 Week, 6 Months, 1 Year).
- Changing the selection automatically updates the chart.

Chart Section:

- Shows historical prices, backtest results, and forecasted prices in an interactive, responsive chart using Chart.js.
- Maintains the dark theme consistent with the website's design.

Data Update: Automatically update forecast data every Monday through Friday at 7:30 PM Bangkok

Compare Models Page



Template Inheritance: Inherits from base.html to reuse the layout, navbar, flash messages, and global styles without duplicating code.

Page Heading: Displays the page title “Compare Models” along with the selected stock symbol, including a link back to the Stock Detail page for easy navigation.

Model Selection Toolbar: Each button links to the Forecasting page for the corresponding model (ARIMA, SARIMA, SARIMAX, LSTM).

Forecast Horizon Filter:

- Dropdown menu to choose the forecast horizon (1 Week, 6 Months, 1 Year).
- Changing the selection refreshes the table and chart automatically.

Summary Card

- Shows the Last Price and the Best Model (most accurate model).
- Uses badges to emphasize the best-performing model.

Comparison Table

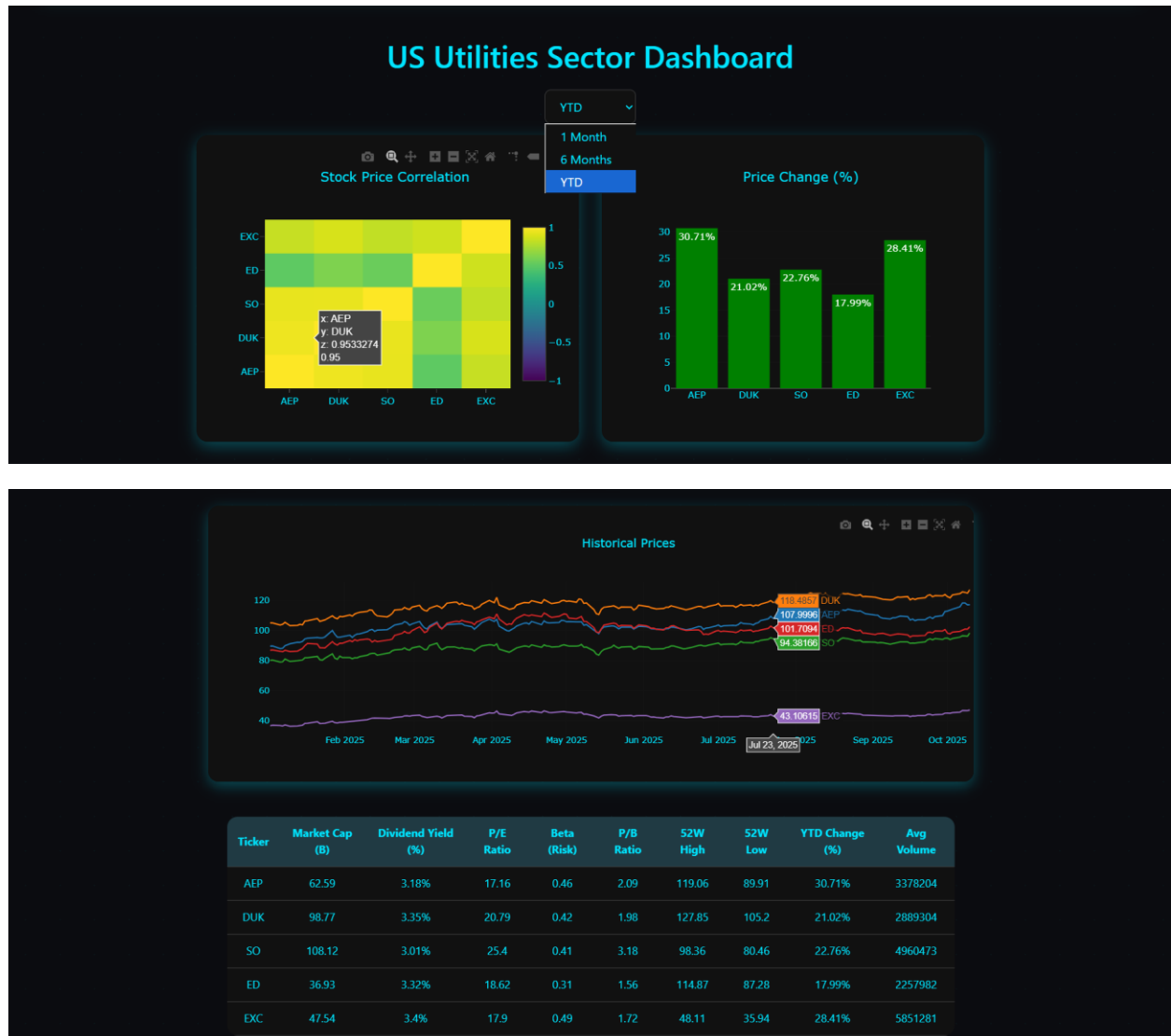
- Displays a table comparing all four models with MAE, %Error, Forecast, and Status.
- Check marks indicate successful models, while ERR badges indicate errors.

Forecast Chart

- Line chart visualizing historical prices and forecasts from each model.
- Chart.js is used for an interactive and color-coded display for each model.

Data Update: Automatically update forecast data every Monday through Friday at 7:30 PM
Bangkok

Dashboard Page



Template Inheritance: Inherits from base.html to reuse the layout, navbar, flash messages, and global styles without duplicating code.

Page Heading: Displays the title “US Utilities Sector Dashboard” to indicate the overall view of the sector.

Time Filter Dropdown:

- Allows users to select a time range (1 Month, 6 Months, YTD).
- The selected range dynamically updates charts to reflect the chosen period.

Correlation & Price Change Charts:

- Correlation Heatmap: Shows the price correlation between each stock.
- Price Change Bar Chart: Displays the percentage change of each stock. Colors (green/red) indicate price increase or decrease for quick visual interpretation.

Historical Prices Chart:

- Line chart showing historical prices of all stocks within the selected time range.
- Helps users visualize long-term trends and patterns for each stock.

Stocks Data Table:

- Tabular view of key stock metrics: Market Cap, Dividend Yield, P/E Ratio, Beta, P/B Ratio, 52W High/Low, YTD Change, and Avg Volume.
- Uses background color coding per row for better visual distinction.